Online machine learning with decision trees

Max Halford

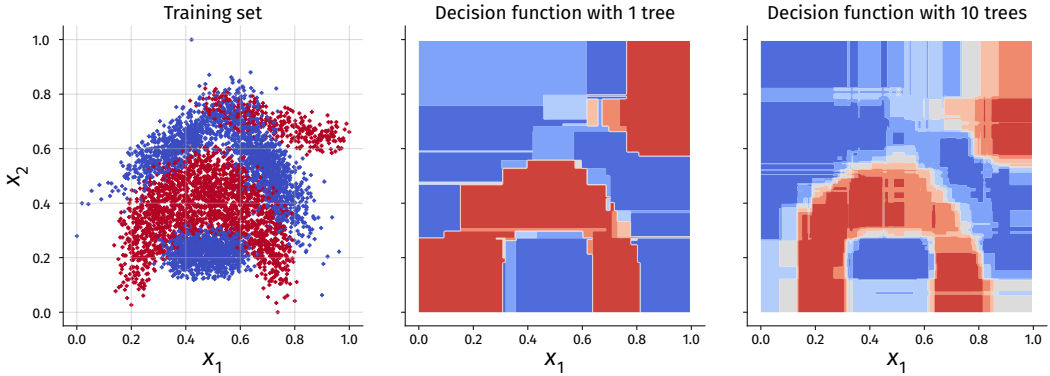
University of Toulouse
Thursday 7th May, 2020

Decision trees

- "*Most successful general-purpose algorithm in modern times.*" [HB12]
- Sub-divide a feature space into partitions
- Non-parametric and robust to noise
- Can be regularised in many ways
- Good weak learners for bagging and boosting
- Many popular open-source implementations [PVG+11, CG16, KMF+17, PGV+18]

But, they can't be trained online, because they assume that the data can be scanned multiple times.

Toy example: the banana dataset [1]



[1] Banana dataset on OpenML

Online (supervised) machine learning

- Model learns from samples $(x, y) \in \mathbb{R}^{n \times p} \times \mathbb{R}^{n \times k}$ which arrive in sequence
- Online != out-of-core:
  - Online: samples are only seen once
  - Out-of-core: samples can be revisited
- **Progressive validation** [BKL99]: $\hat{y}$ can be obtained right before y is shown to the model, allowing the training set to also act as a validation set. No need for cross-validation!
- Ideally, **concept drift** [GŽB+14] should be taken into account:
  1. **Virtual drift**: $P(X)$ changes
  2. **Real drift**: $P(Y \mid X)$ changes:
     - Example: many 0s with sporadic bursts of 1s
     - Example: a feature's importance changes through time

Online decision trees

- A decision tree involves enumerating split candidates
- Each split is evaluated by scanning the data
- This can't be done online without storing data
- Two approaches to circumvent this:
  1. Store and update feature distributions
  2. Build the trees without looking at the data (!!)
- Bagging and boosting can be done online [OR01]

Consistency

- Trees fall under the non-parametric regression framework
- Goal: estimate a regression function $f(x) = \mathbb{E}(Y \mid X = x)$
- We estimate f with an approximation f_n trained with n samples
- f_n is consistent if $\mathbb{E}(f_n(X) - f(X))^2 \to 0$ as $n \to +\infty$
- Ideally, we also want our estimator to be unbiased
- We also want regularisation mechanisms in order to generalise

Hoeffding trees (1)

- Split thresholds t are chosen by minimising an impurity criterion
- An impurity criterion depends on $P(Y \mid X < t)$, which can be obtained via Bayes' rule:

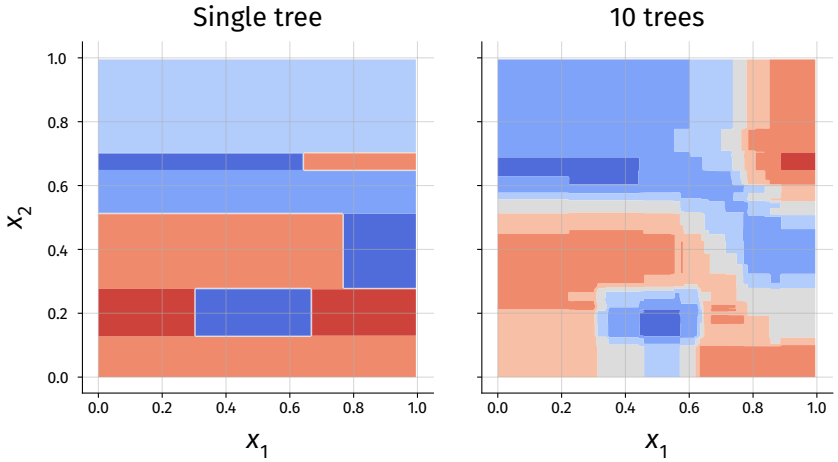
$$P(Y \mid X < t) = \frac{P(X < t \mid Y) \times P(Y)}{P(X < t)}$$

- For classification, assuming X is numeric:
  - P(Y) is a counter
  - P(X < t) can be represented with a histogram
  - P(X < t ? Y) can be represented with one histogram per class

Hoeffding trees (2)

- A Hoeffding tree starts off as a leaf
- $P(Y)$, $P(X < t)$, and $P(X < t \mid Y)$ are updated every time a sample arrives
- Every so often, we enumerate some candidate splits and evaluate them
- The best split is chosen if significantly better than the second best split
- Significance is determined by the Hoeffding bound
- Once a split is chosen, the leaf becomes a branch and the same steps occur within each child
- Introduced in [DH00]
- Many variants, including revisiting split decisions when drift occurs [HSD01]

Hoeffding trees on the banana dataset



Mondrian trees

- Construction follows a Mondrian process [RT+08]
- Split features and points are chosen without considering their predictive power
- Hierarchical averaging is used to smooth leaf values
- First introduced in [LRT14]
- Improved in [MGS19]

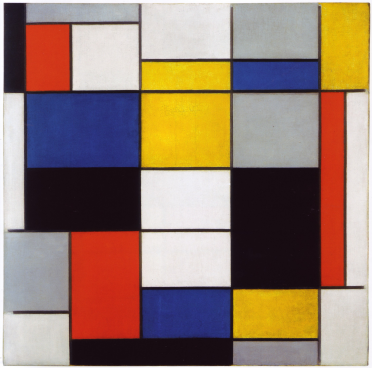
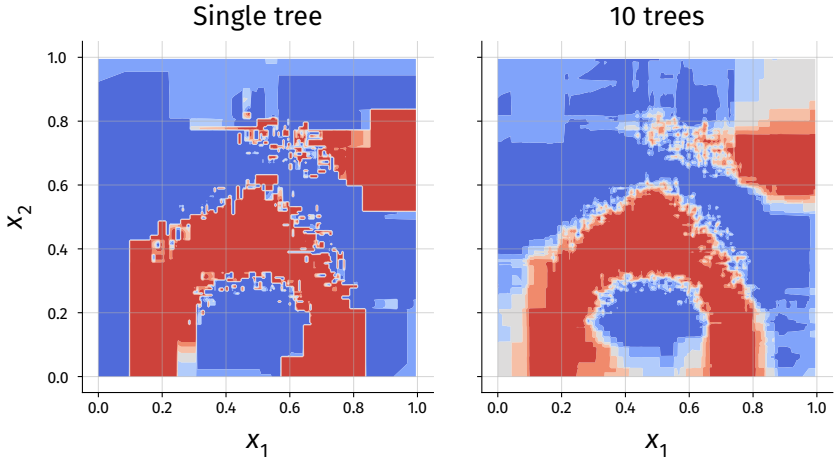


Figure: *Composition A* by Piet Mondrian

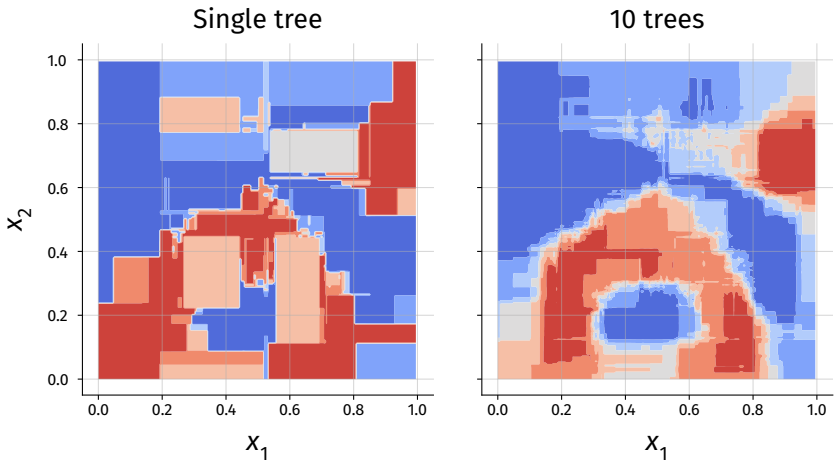
The Mondrian process

- Let u_j and l_j be the bounds of feature j in a cell
- Sample $\delta \sim exp(\sum_{j=1}^{p} u_j - l_j)$
- Split if $\delta < \lambda$
- The chances of splitting decrease with the size of the cells
- λ is a soft maximum depth parameter
- Features are uniformly chosen in proportion to $u_j - l_j$
- More information in these slides

Mondrian trees on the banana dataset



Aggregated Mondrian trees on the banana dataset



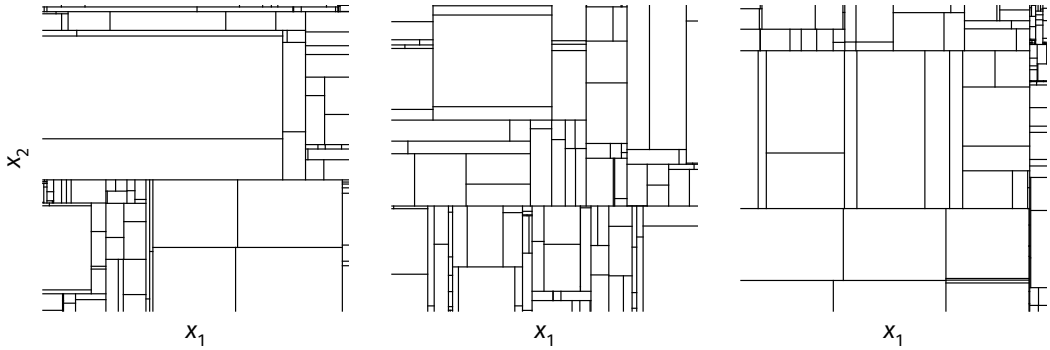
Purely random trees

- Features x are assumed to in $[0, 1]^p$
- Trees are constructed independently from the data, before it even arrives:
  1. Pick a feature at random
  2. Pick a split point at random
  3. Repeat until desired depth is reached
- When a sample reaches a leaf, said leaf's running average is updated
- Easier to analyse because tree structure doesn't depend on Y
- Consistency depends on:
  1. The height of a tree – denoted h
  2. The amount of features that are "relevant"

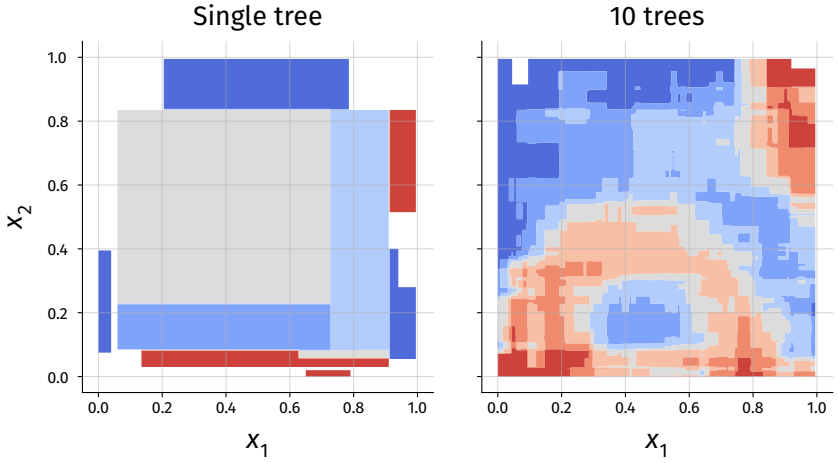
Uniform random trees

- Features and split points are chosen completely at random
- Let h be the height of the tree
- Consistent when $h \to +\infty$ and $\frac{h}{n} \to 0$ as $h \to +\infty$ [BDL08]

Uniform random trees



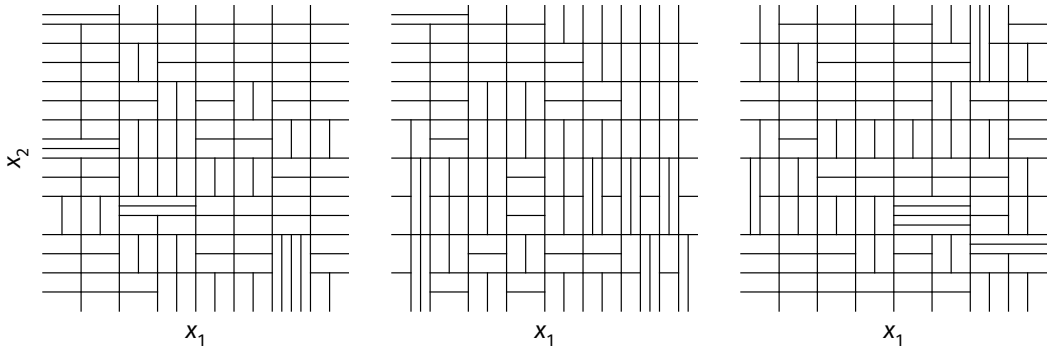
Uniform random trees on the banana dataset



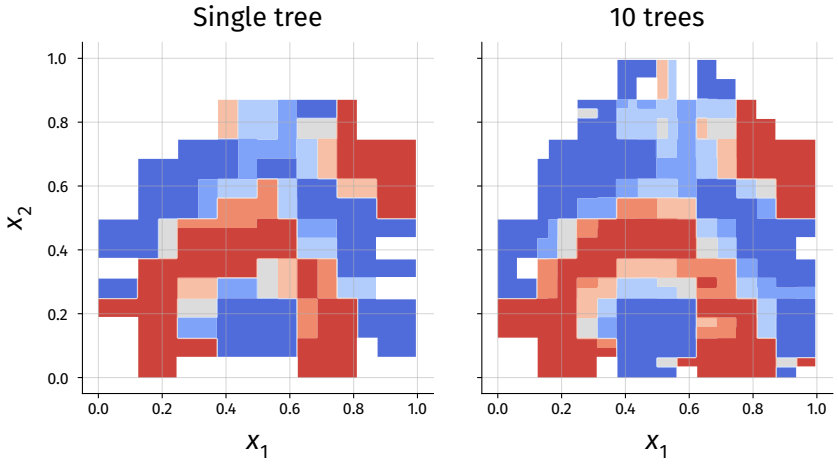
Centered random trees

- Features are chosen completely at random
- Split points are the mid-points of a feature's current range
- Consistent when $h \to +\infty$ and $\frac{n}{2^h} \to 0$ as $h \to +\infty$ [Sco16]

Centered random trees

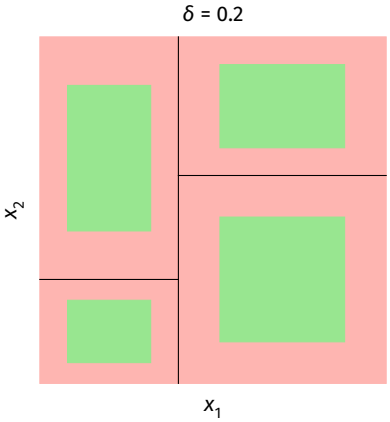


Centered random trees on the banana dataset

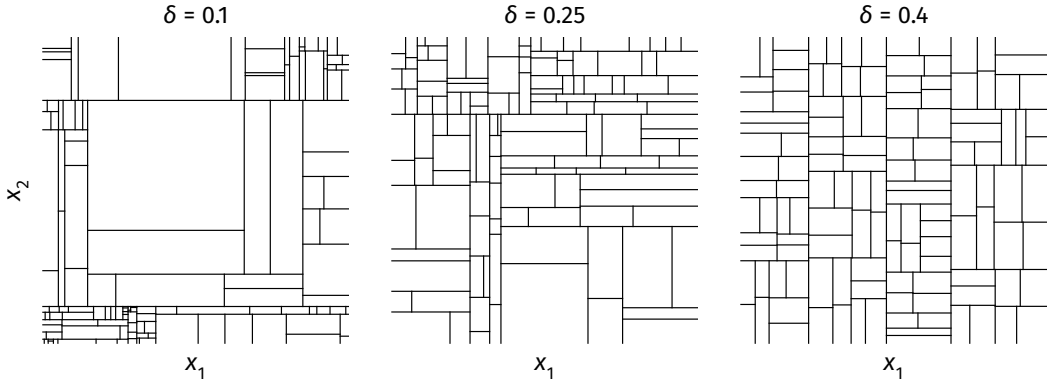


How about a compromise?

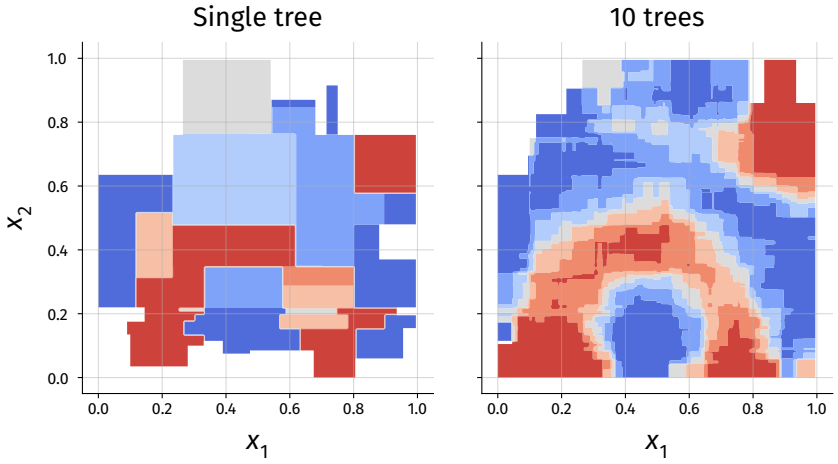
- Choose $\delta \in [0, \frac{1}{2}]$
- Sample s in $[a + \delta(b - a), b - \delta(b - a)]$
- $\delta = 0 \implies s \in [a, b]$ (uniform)
- $\delta = \frac{1}{2} \implies s = \frac{a+b}{2}$ (centered)



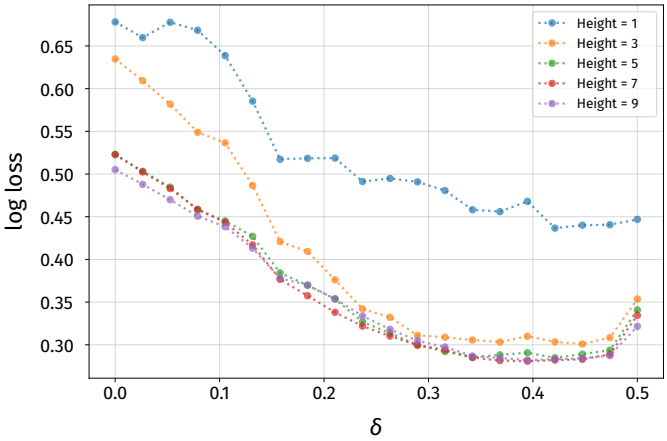
Some examples



Banana dataset with $\delta = 0.2$



Impact of padding on performance



Tree regularisation

- A decision tree overfits when it's leaves contain too few samples
- There are many popular ways to regularise trees:
  1. Set a lower limit on the number of samples in each leaf
  2. Limit the maximum depth
  3. Discard irrelevant nodes after training (pruning)
- We can't do any of this in an online learning context :(

Hierarchical smoothing

- Intuition: a leaf doesn't contain enough samples... but it's ancestors might!
- Let $G(x_t)$ be the nodes that go from the root to the leaf for a sample x_t
- **Curtailment** [ZE01]: use the first node in $G(x_t)$ with at least k samples
- Aggregated Mondrian trees [MGS19] use context weighting trees

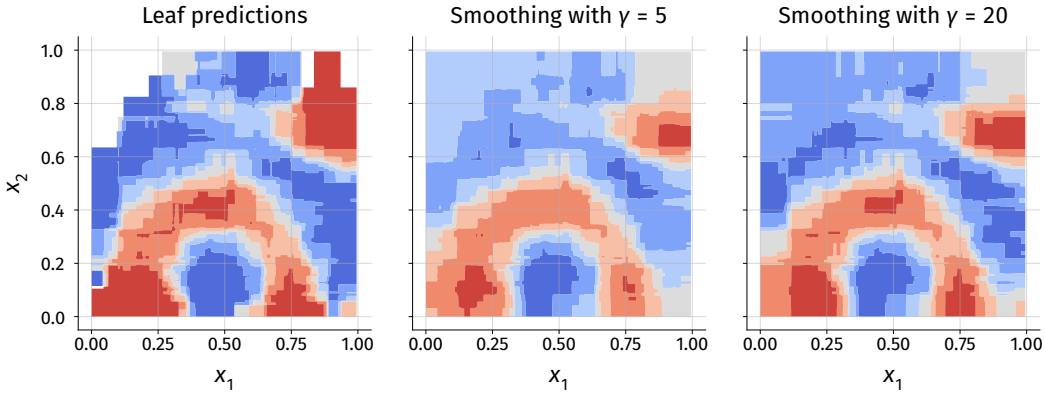
A simple averaging scheme

- Idea: make each node in $G(x_t)$ contribute to a weighted average
- Let,
  - k be the number of samples in a node
  - d be the depth of a node
- Then, the contribution of each node can be weighted by:

$$w = k \times (1 + \gamma)^d$$

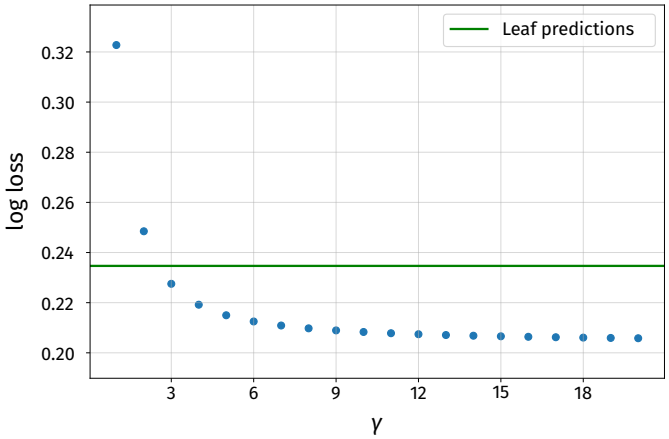
- The more samples a leaf contains, the more it matters
- The deeper a leaf is, the more it matters
- $\gamma \in \mathbb{R}$ controls the importance of both values
- I like to call this **path averaging**

Averaging on the banana dataset



Notice what happens in the corners.

Impact of γ on predictive performance



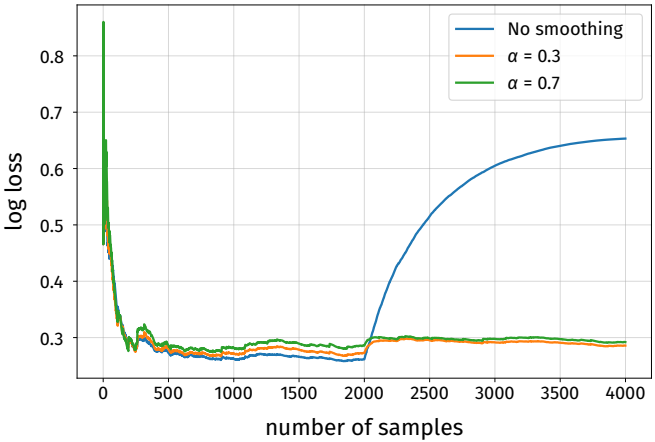
Dealing with concept drift

- Each node contains a running average of the y values it has seen
- Instead, we can maintain an exponentially weighted moving average (EWMA):

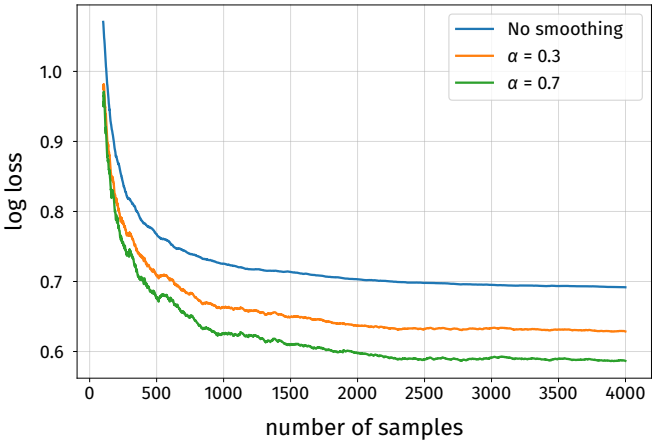
$$\bar{y}_t = \alpha y_t + (1 - \alpha)\bar{y}_{t-1}$$

- α determines the influence of the most recent values

Hard drift: flip y values after 2000 samples



Soft drift: slowly rotate samples around barycenter



Feature selection

A limitation of AMF, however, is that it does not perform feature selection. It would be interesting to develop an online feature selection procedure that could indicate along which coordinates the splits should be sampled in Mondrian trees, and prove that such a procedure performs dimension reduction in some sense. This is a challenging question in the context of online learning which deserves future investigations.

Final paragraph from [MGS19]

Online feature selection is a difficult problem!

A solution?

1. Initially, we don't know the importance of each feature, so we pick them at random
2. After some time, we can measure the quality of each split within each tree
3. We can derive the feature importances from the splits each feature participates in
4. Every so often we can build a new tree by sampling feature relative to their importances
5. The selection probabilities should be conditioned on the features already chosen

This is still work in progress, but there is hope.

Parameters recap

- m: number of trees
- h: height of each tree
- δ: the amount of padding
- γ: determines how the path averaging works
- α: exponentially weighted moving average parameter

Some useful Python libraries

- `scikit-garden` – Mondrian trees
- `onelearn` – Aggregated Mondrian trees
- `scikit-multiflow` – Hoeffding trees
- `scikit-learn` – General-purpose batch machine learning
- `creme` – General-purpose online machine learning

Train/test benchmarks

	Moons	Noisy linear	Higgs	Higgs*
Batch log reg	.324	.244	.640	.677
Batch log reg with Fourier features	.193	.213	.698	.641
Batch random forest	.225	.210	.615	.639
NN with 2 layers	.171	.196	.653	.637
Online log reg	.334	.323	.662	.677
Mondrian forest	.349	.316	.692	.905
Aggregated Mondrian forest	.205	.199	.671	.649
Hoeffding forest	.330	.258	.664	.649
Padded trees (us)	.185	.193	.678	.644

Streaming benchmarks

Work in progress!

Feedback is more than welcome. Thanks for listening.

References

Gérard Biau, Luc Devroye, and Gábor Lugosi.
Consistency of random forests and other averaging classifiers.
Journal of Machine Learning Research, 9(Sep):2015–2033, 2008.

Avrim Blum, Adam Kalai, and John Langford.
Beating the hold-out: Bounds for k-fold and progressive cross-validation.
In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208, 1999.

Tianqi Chen and Carlos Guestrin.
Xgboost: A scalable tree boosting system.
In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

References

Pedro Domingos and Geoff Hulten.
Mining high-speed data streams.
In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia.
A survey on concept drift adaptation.
ACM computing surveys (CSUR), 46(4):1–37, 2014.

Jeremy Howard and Mike Bowles.
The two most important algorithms in predictive modeling today.
In *Strata Conference presentation, February*, volume 28, 2012.

References

Geoff Hulten, Laurie Spencer, and Pedro Domingos.
Mining time-changing data streams.
In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu.
Lightgbm: A highly efficient gradient boosting decision tree.
In *Advances in neural information processing systems*, pages 3146–3154, 2017.

Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh.
Mondrian forests: Efficient online random forests.
In *Advances in neural information processing systems*, pages 3140–3148, 2014.

References

- Jaouad Mourtada, Stéphane Gaïffas, and Erwan Scornet.
  Amf: Aggregated mondrian forests for online learning.
  arXiv preprint arXiv:1906.10529, 2019.
- Nikunj Chandrakant Oza and Stuart Russell.
  Online ensemble learning.
  University of California, Berkeley, 2001.
- Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin.
  Catboost: unbiased boosting with categorical features.
  In *Advances in neural information processing systems*, pages 6638–6648, 2018.

References

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.
Scikit-learn: Machine learning in Python.
Journal of Machine Learning Research, 12:2825–2830, 2011.

Daniel M Roy, Yee Whye Teh, et al.
The mondrian process.
In *NIPS*, pages 1377–1384, 2008.

Erwan Scornet.
On the asymptotics of random forests.
Journal of Multivariate Analysis, 146:72–83, 2016.

References

Bianca Zadrozny and Charles Elkan.
Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers.
In *Icml*, volume 1, pages 609–616. Citeseer, 2001.